

...we have not mastered the art of developing error free SW! (... and probably never will!)

- “error free” SW subject to errors from the environment
- ▶ Defensive SW outlook to minimize impact of **data errors**:

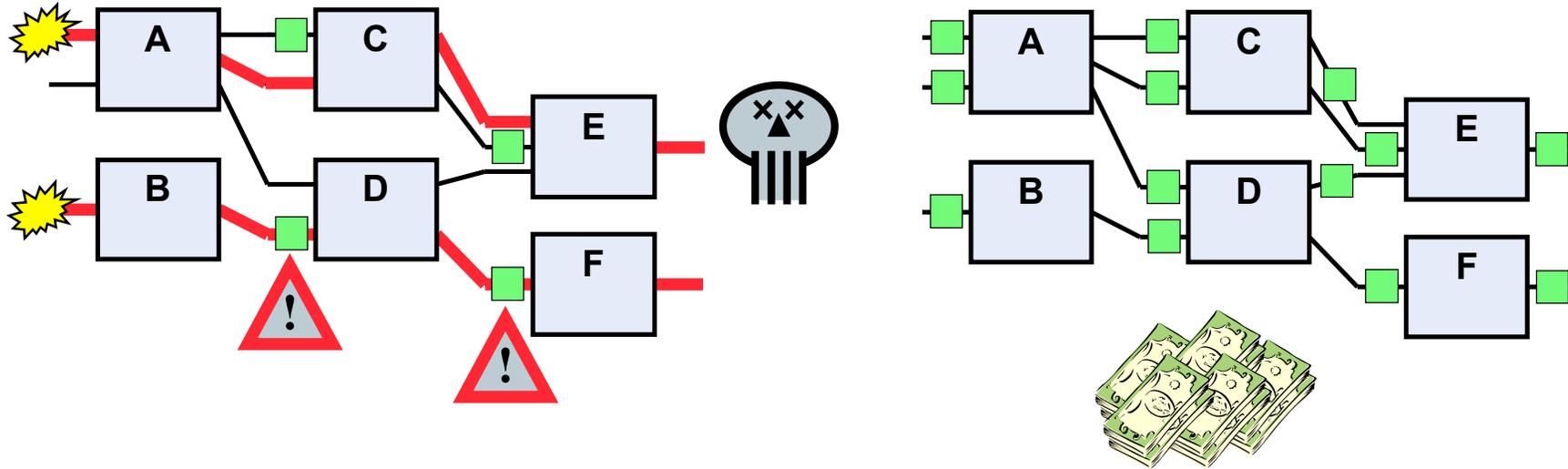
(error firewalling)

- at the code/module level (leaky, buggy SW)
- at the OS module level to prevent “data errors” from an application or user to impact the OS functionality

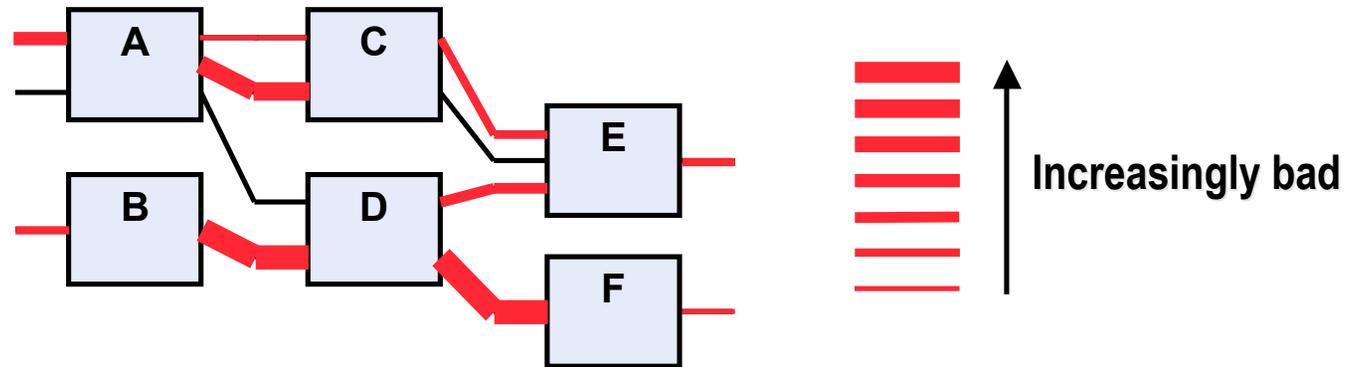
**if** DATA\_ERROR = ... **then** ... (1/x ; if x =0, then ...)

**if** ... **then** → exception detection and handling  
*Executable Assertions (EA's)*

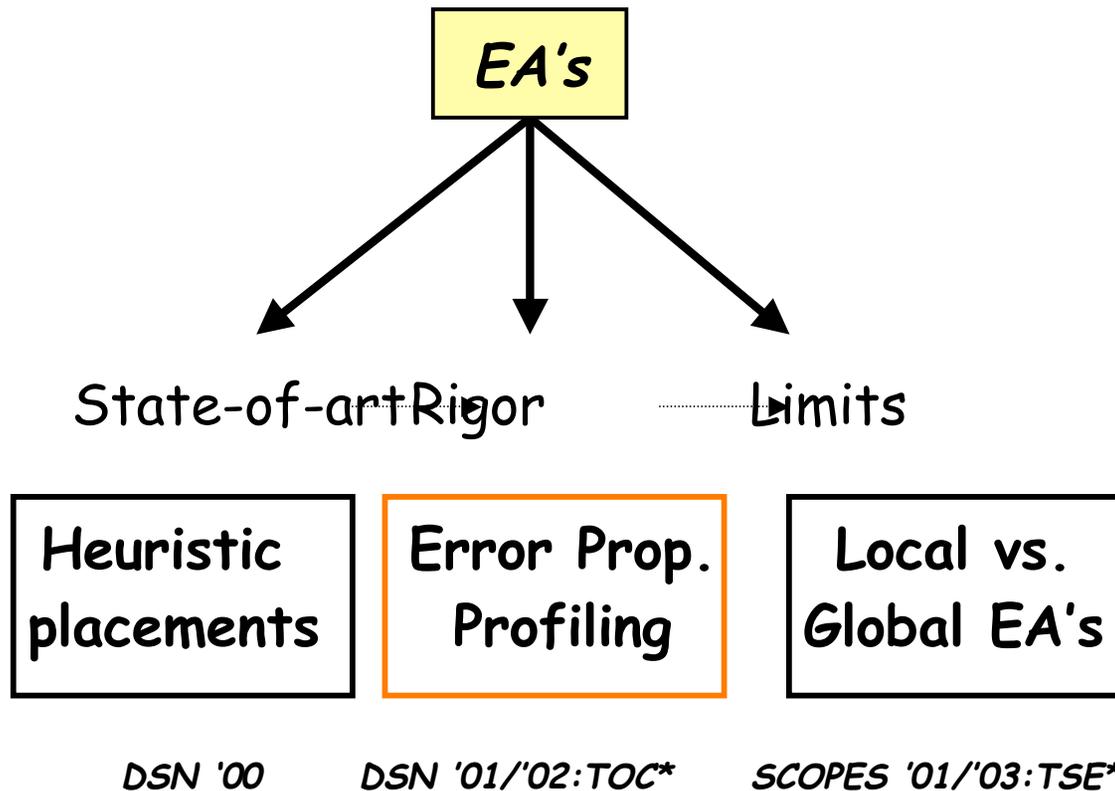
# EA Placements: Motivating Software Profiling



Obtain profiles of the software system to select EA locations.



# Was, Is, Will be ...



# The Art & Science of Error Containment in SW: Profiling Error Propagation

Neeraj Suri

TU Darmstadt, Germany / U. Texas, Austin

<http://www.deeds.informatik.tu-darmstadt.de>

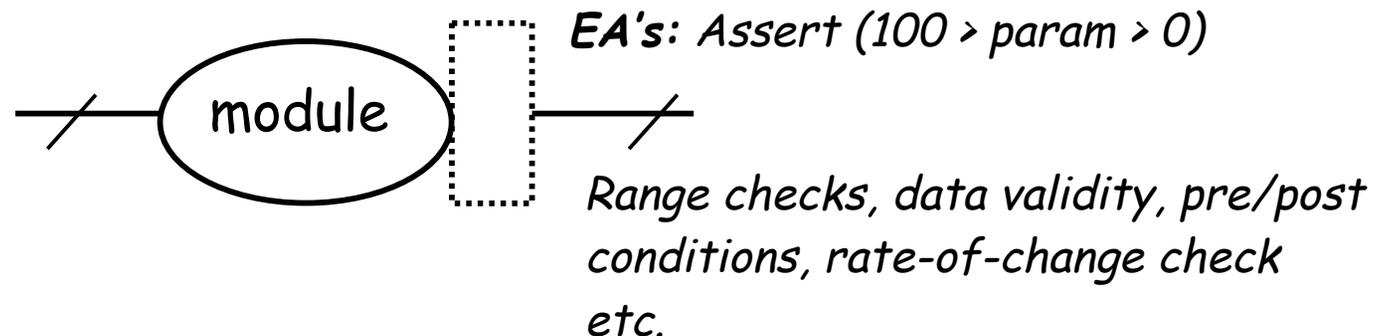


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Pragmatism → Greybox SW

Pre-constructed SW modules (objects):

- Basic functional & I/O information known
- Internals not (fully) known or changeable
- Additions of EA's/wrappers/constraint checkers possible



EA's: (if-then) code fragments checking validity of system state (signals/variables)

## Objectives & Outline

Enhancing Robustness of Embedded SW using Wrappers -- Given a SW system (and resource constraints):

- How to profile flow of (data) errors in SW?
- How to effectively locate & utilize EA's (EDM + ERM's)  
(EDM = Error Detection , ERM = Error Recovery Mechanisms)
  - ▶ Design time cost-coverage tradeoff guidance!
- EA's in a conformal distributed sense/EA Composition?
- ... and in a quantifiable, reproducible manner!!!

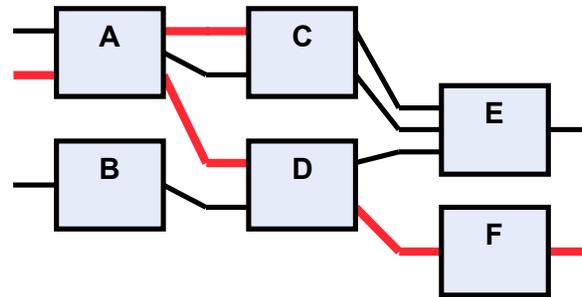
# Profiling Modular Software

Software profiling w.r.t.

Propagation

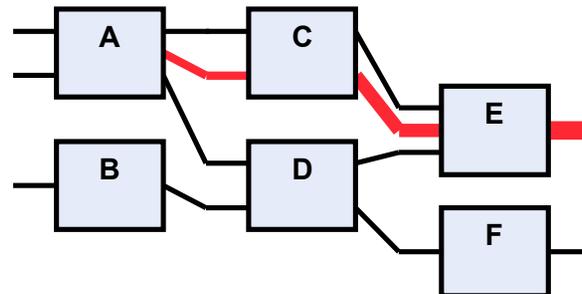
appears?

**WHERE** does an error **GO** when it



Effect

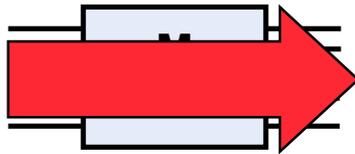
**WHAT** does an error **DO** when it appears?



## Basis: Error Propagation & Effect Analysis

- Assess the propagation of *data errors* in modular software
  - Find module's ability to pass errors (*Error Permeability*)
  - Find module/signal exposure to errors (*Error Exposure*)
  - Determine effect of errors (*Error Impact/Criticality*)
- Locate EDM's and ERM's

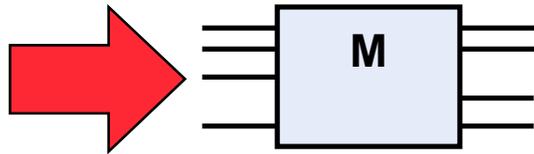
# SW Profiling: Propagation Analysis of Data Errors



## Module Error Permeability

To what degree does a module let errors "pass through"

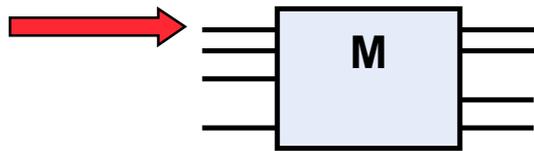
High value → error containment should be increased



## Module Error Exposure

To what degree is a module "exposed" to propagating errors

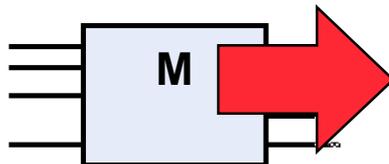
High value → error shielding should be high



## Signal Error Exposure

To what degree is a signal "exposed" to propagating errors

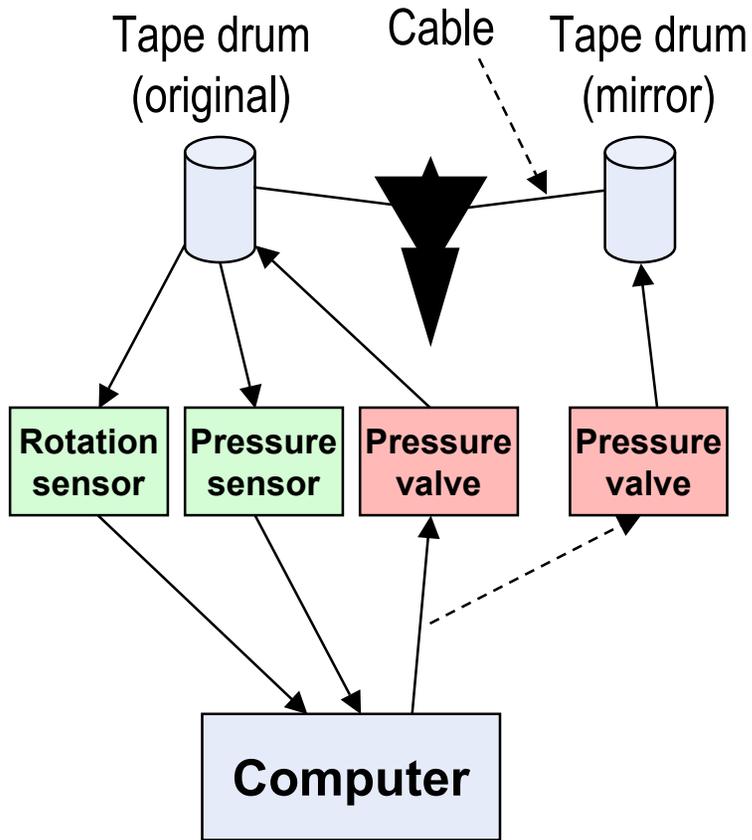
High value → error shielding should be high



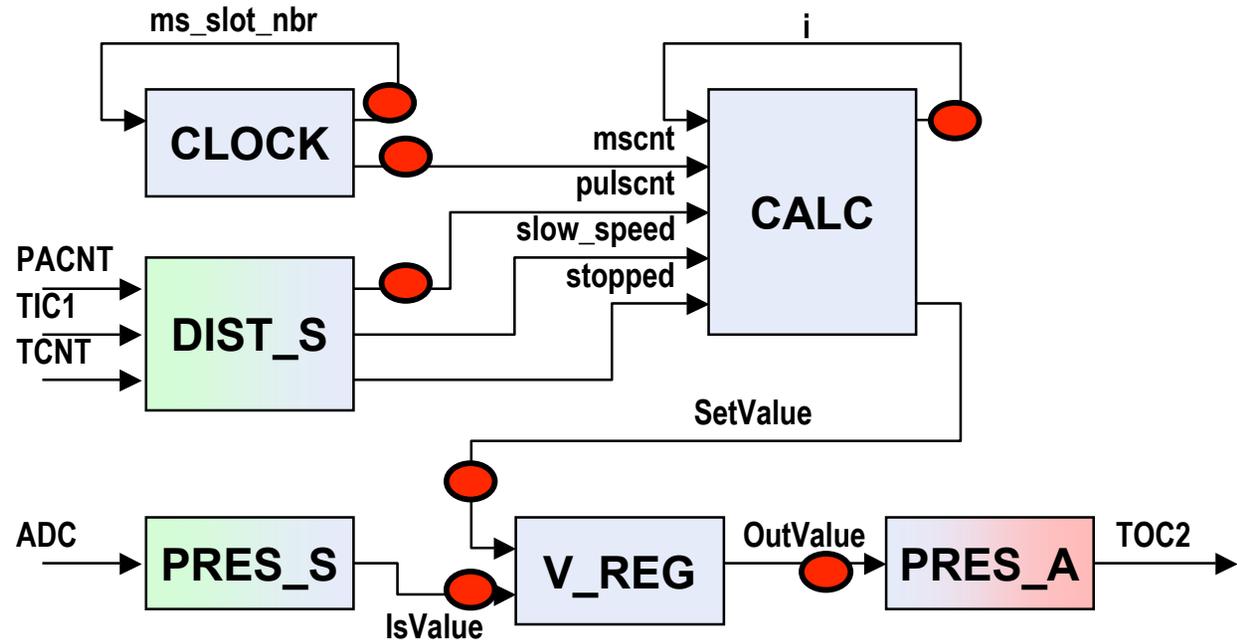
## *Error Impact/Criticality*

What effects/damage can errors potentially cause?

# Example: Aircraft Arrestment System



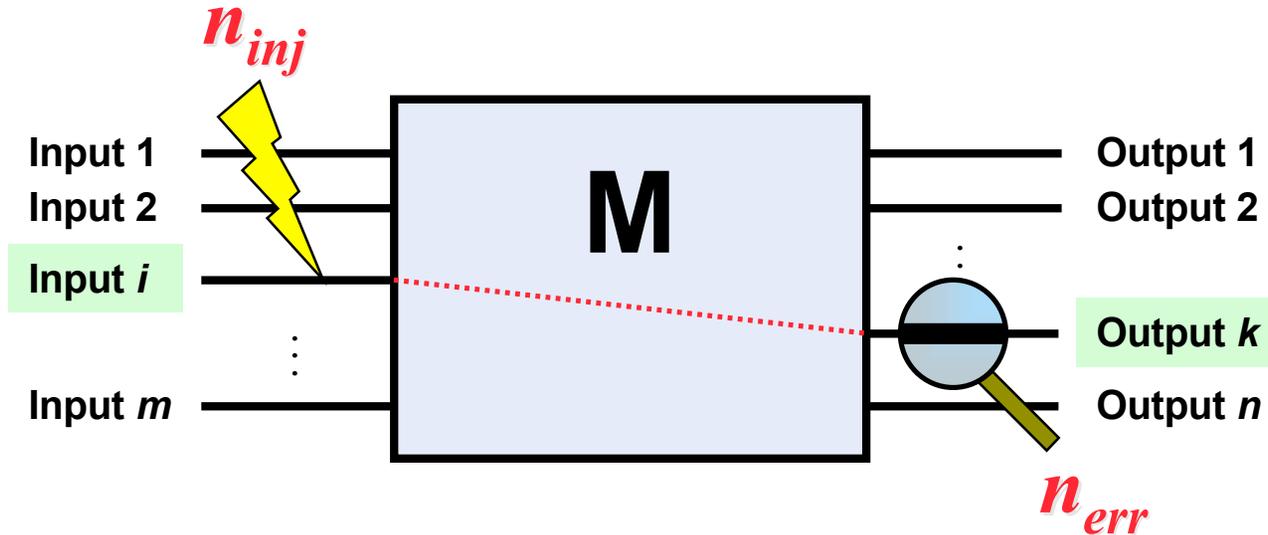
Target system overview



Target software overview

# Error Permeability - Module's Ability to Pass Errors Through

Experimental estimation using code propagation analysis (PROPANE)



Estimated error permeability:

$$P_{i,k}^M = \frac{n_{err}}{n_{inj}}$$

A total of  $m \cdot n$  values for Module M

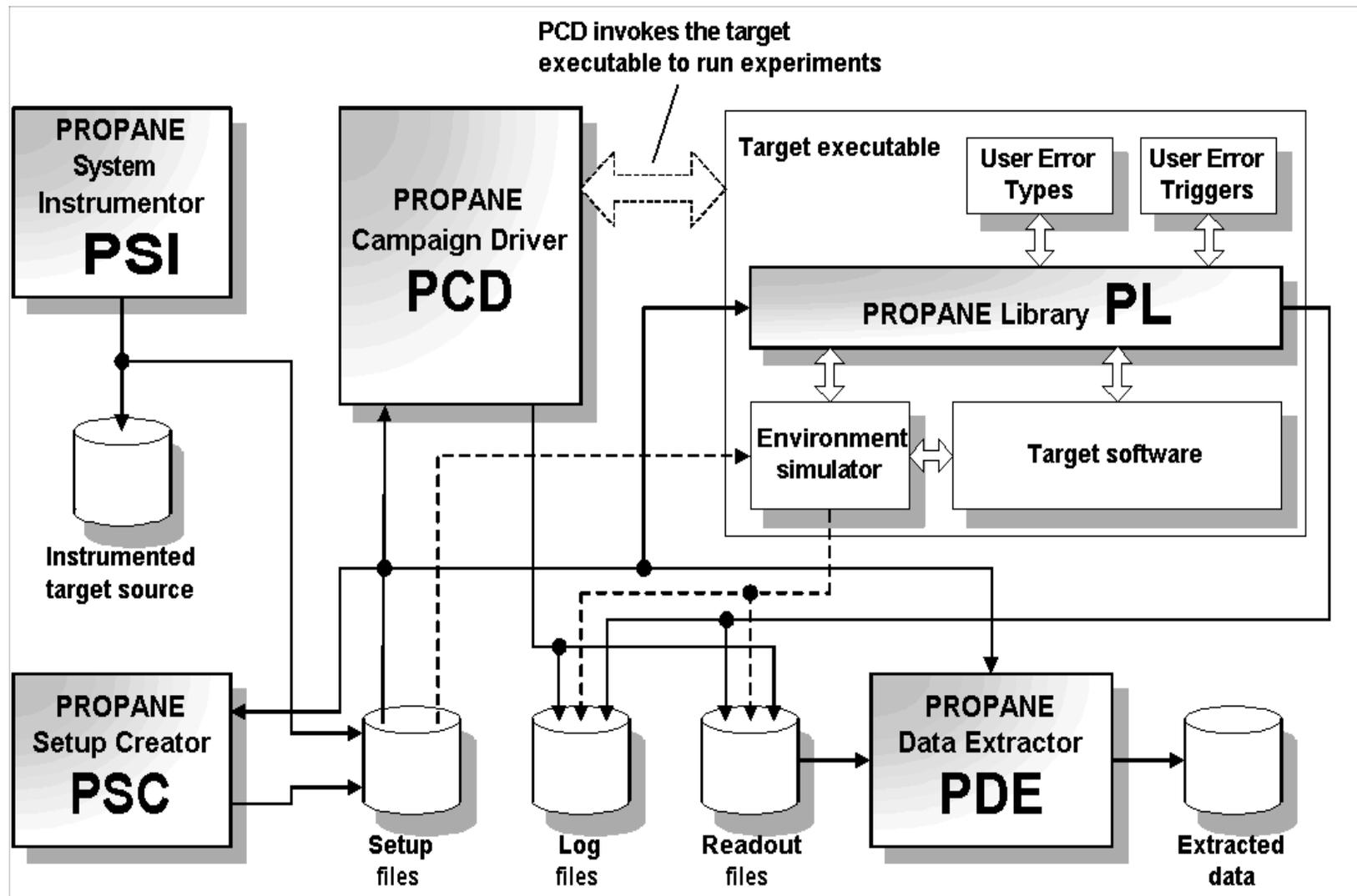
Analytical error permeability: (Basic direct conditional only)

$$0 \leq P_{i,k}^M = \Pr \{ \text{error in output } k \mid \text{error in input } i \} \leq 1$$

Permeability

$$0 \leq \hat{P}^M = \sum_i \sum_k P_{i,k}^M \leq m \cdot n$$

# PROPANE



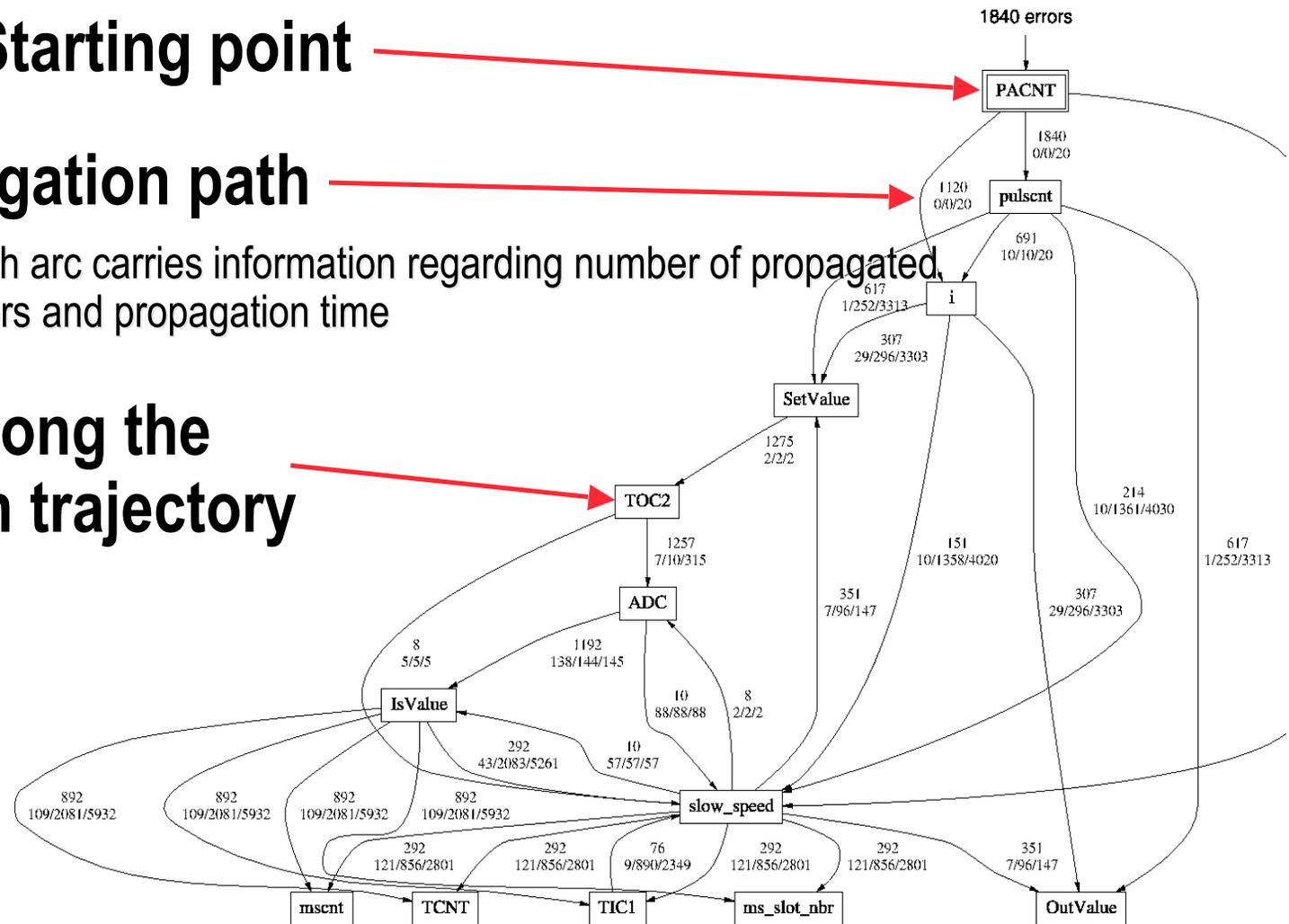
# Propagation Graph for PACNT

Starting point

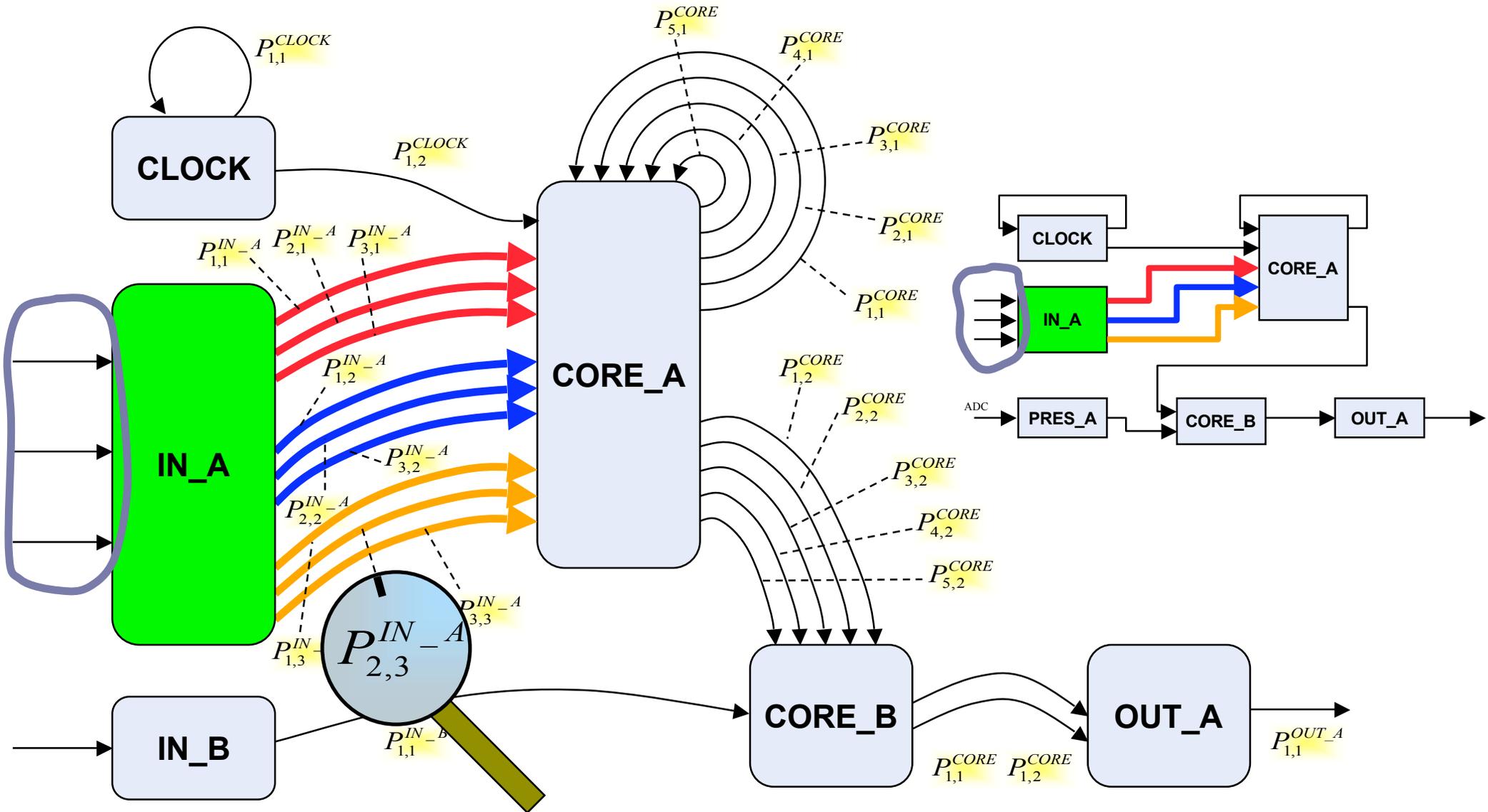
Propagation path

Each arc carries information regarding number of propagated errors and propagation time

Variables along the propagation trajectory



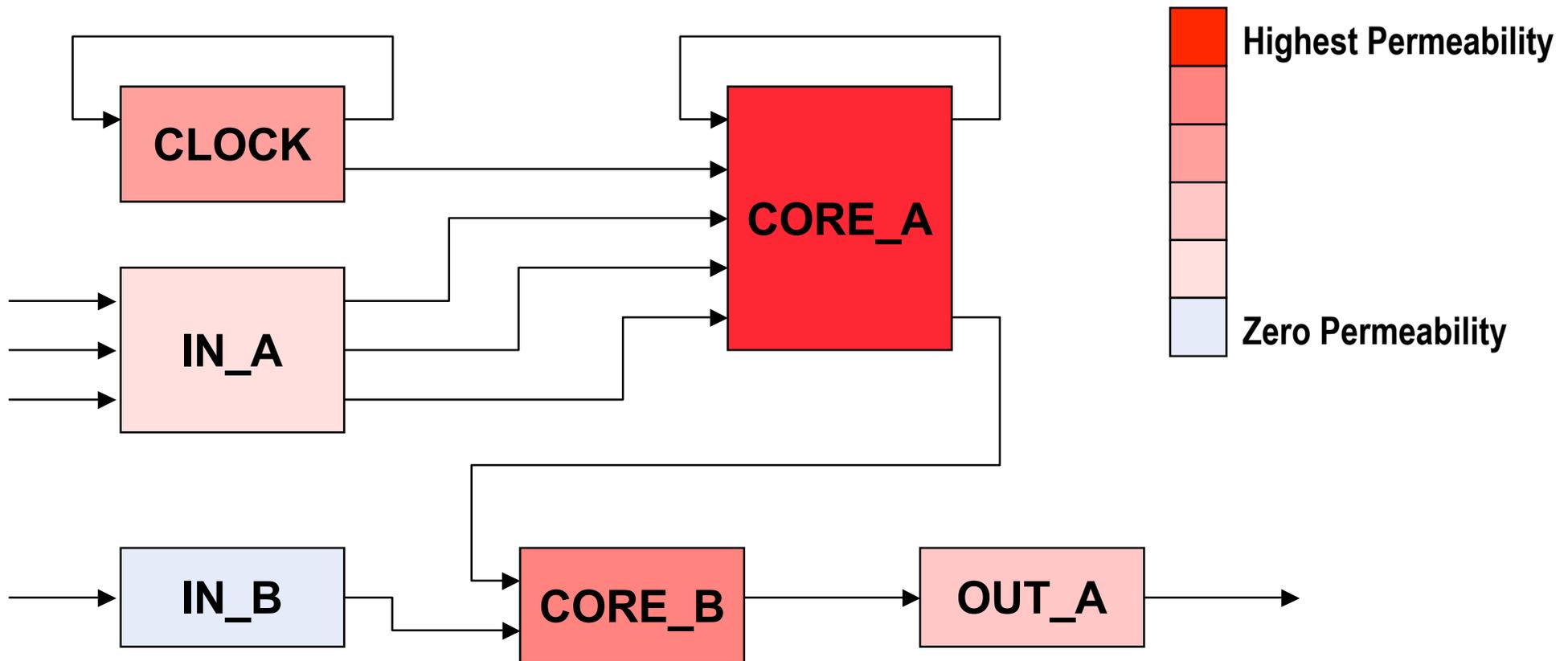
# Permeability Profile



## Guide to Interpreting the Metric

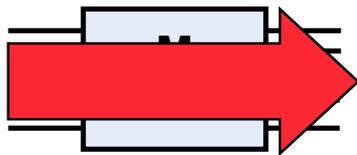
- Higher *permeability* implies higher probability of a module letting errors pass through it
  - Error containment needs to be increased
  - May be cost effective to place EDM/ERM's here

## Estimated Module Permeabilities



Zero permeability (experimental) for errors in IN\_B

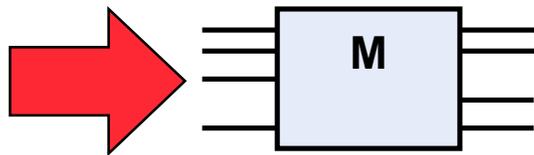
# SW Profiling: Propagation Analysis



## Module Error Permeability

To what degree does a module let errors "pass through"

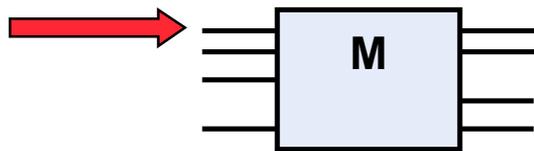
High value → error containment should be increased



## Module Error Exposure

To what degree is a module "exposed" to propagating errors

High value → error shielding should be high

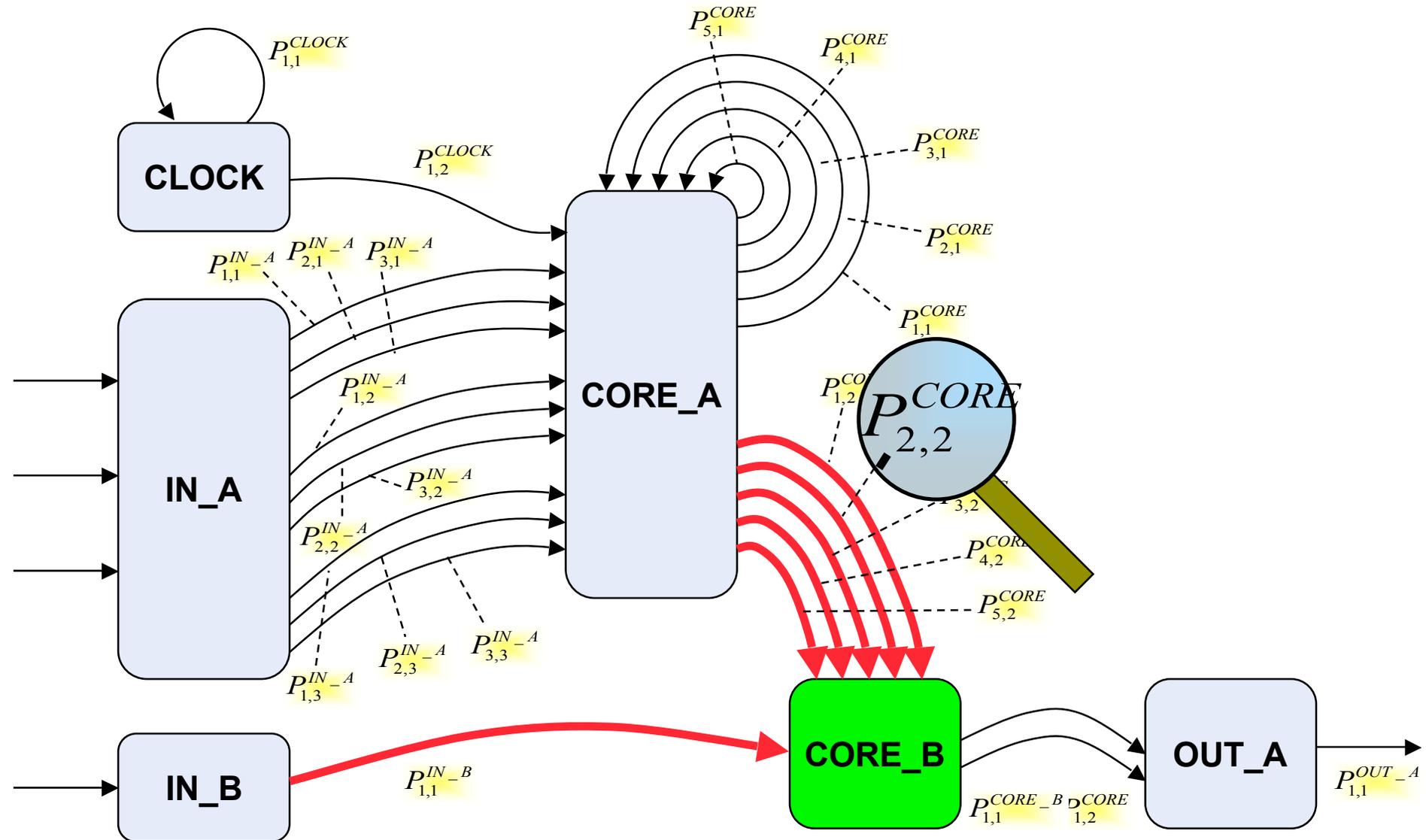


## Signal Error Exposure

To what degree is a signal "exposed" to propagating errors

High value → error shielding should be high

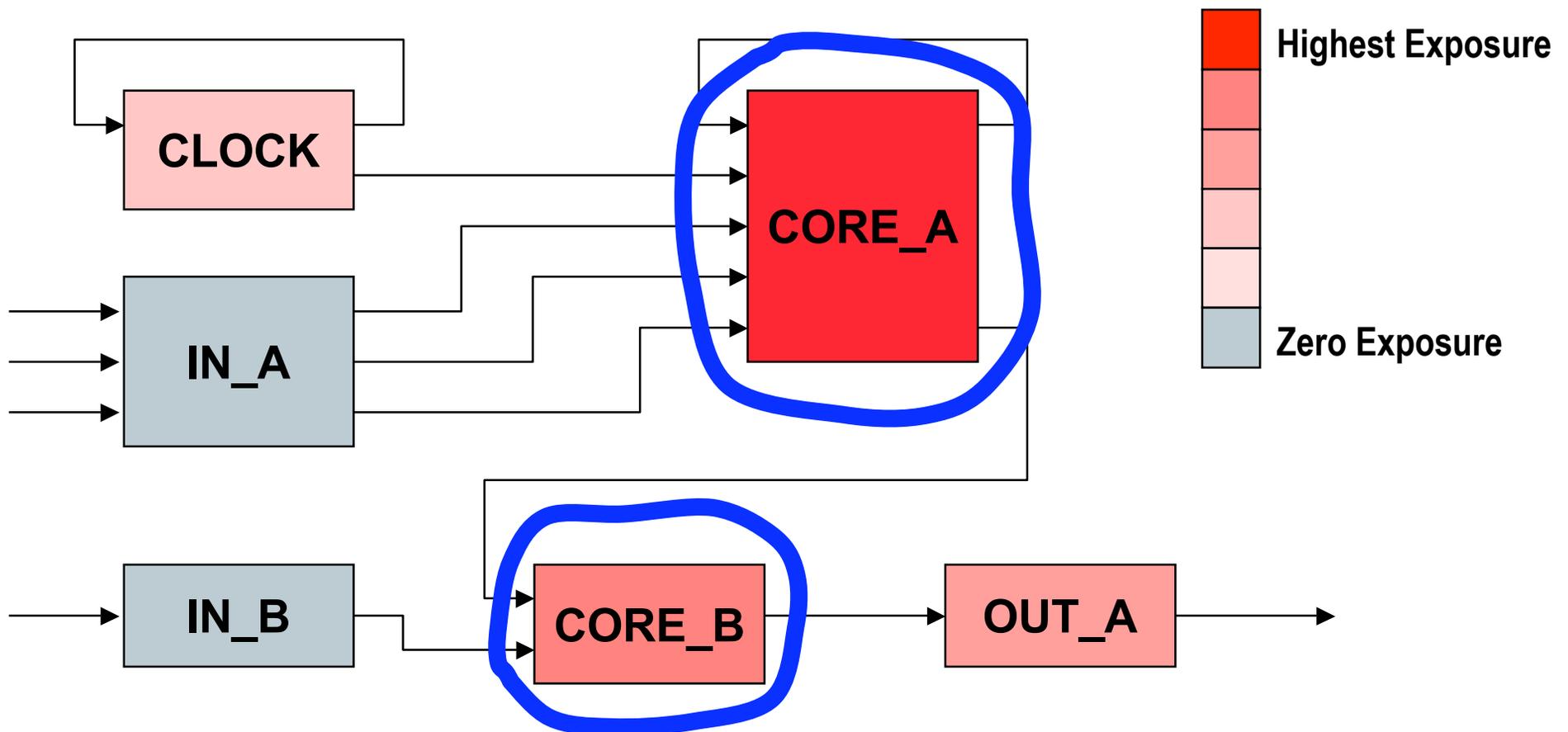
# Error Exposure Profile



## Guide to Interpreting the Metrics

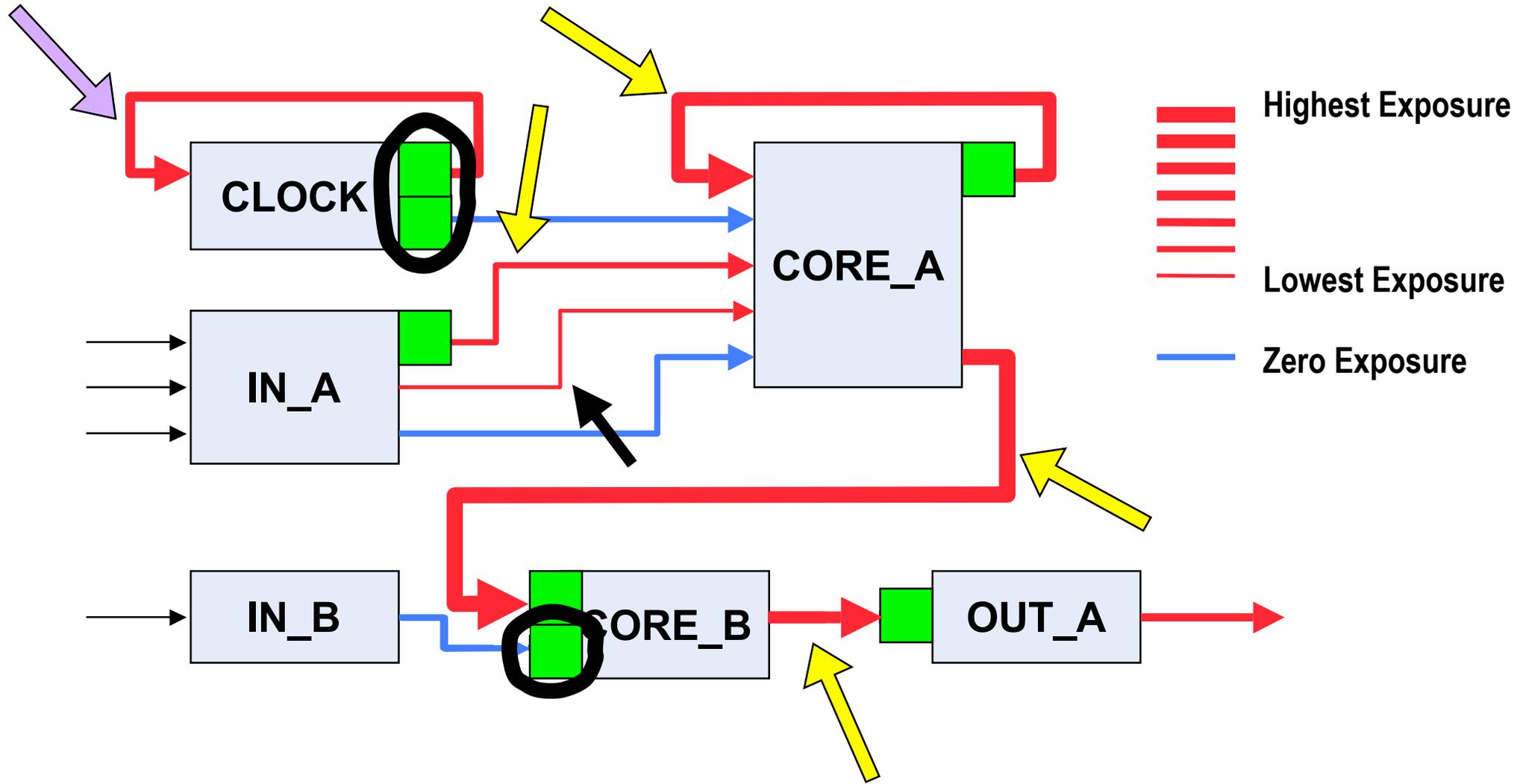
- Higher *exposure* implies higher probability of a module being subjected to propagating errors
  - Central parts of the system that need protection
  - May be cost effective to place EDM's here
- Higher *permeability* implies higher probability of a module letting errors pass through it
  - May be cost effective to place EDM/ERM's here

# Estimated Module Error Exposures



Indicates good candidates for EDM's and ERM's

# Estimated Module → Signal Error Exposures

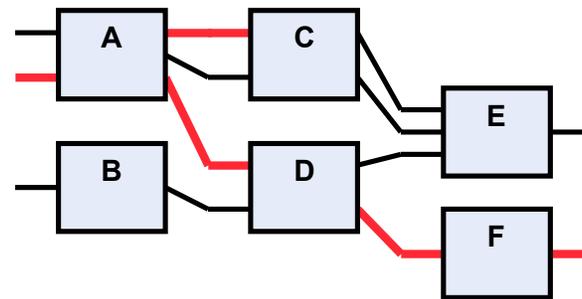


# Profiling Modular Software

Software profiling w.r.t.

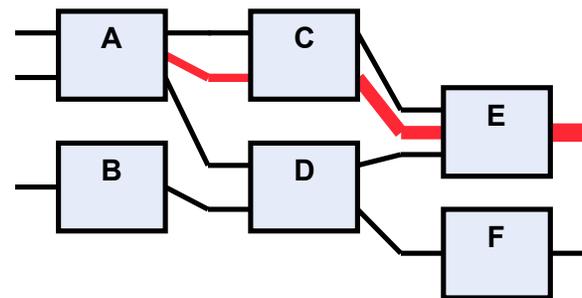
*Propagation*  
appears?

**WHERE** does an error **GO** when it



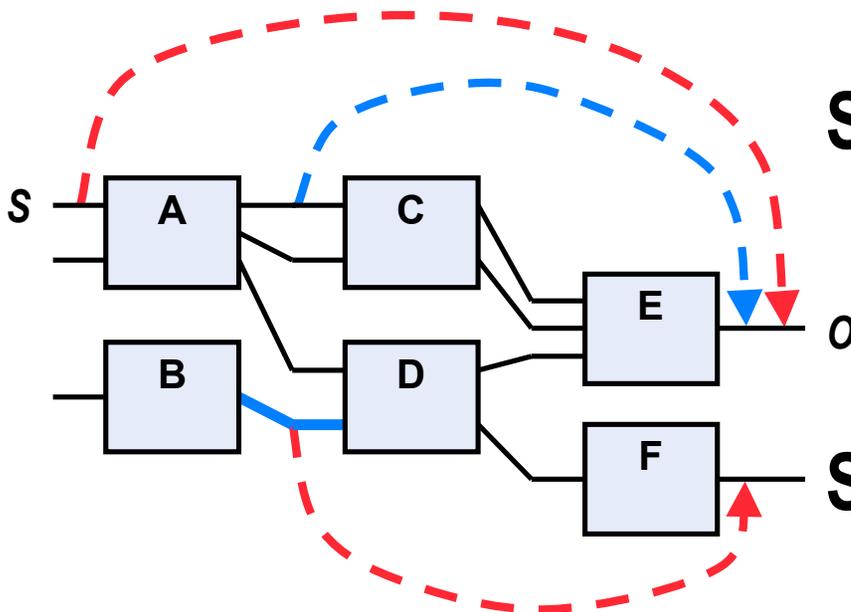
*Effect*

**WHAT** does an error **DO** when it appears?



# SW Profiling: Effect Analysis

Taking care of the *what if*'s



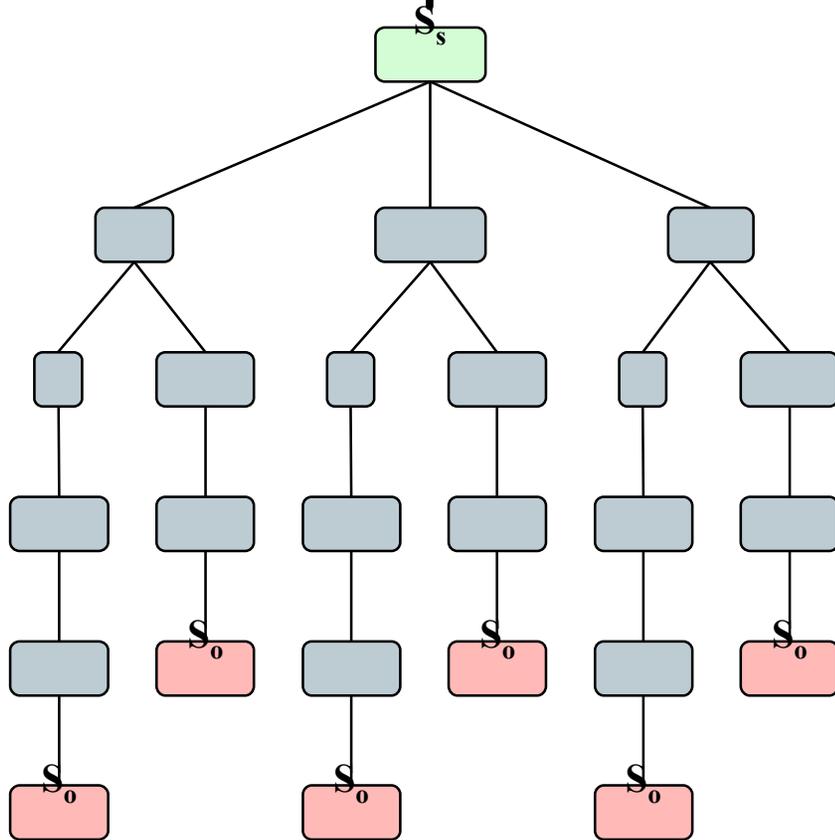
## Signal Impact (of S on O)

To what degree does an error in **S** affect output **O**  
 High value → error recovery should be increased

## Signal Criticality (of S)

"Cost" of an error in **S** as seen from system boundary  
 Multiple outputs: bias *impact* by output "importance"  
 Single output: Constant scaling  
 High value → error recovery should be increased

# Impact - The Effect of Errors



- *Impact* = the effect an error in input signal  $S_s$  has on a system output signal  $S_o$
- *Impact tree* = trace tree = F.tree from  $S_s$  (root) to  $S_o$  (leaves)

**Error Impact of input signal  $S_s$  on system output signal  $S_o$**

$$0 \leq S_s \leq S_o = 1 \prod_i (1 - w_i) \leq 1$$

$w_i$  = weight of path  $i$  (product of permeability values along path)

# Criticality - The Cost of Errors

- ❖ *Criticality* = "cost" of an error in  $S_s$  as seen from system boundary
  - Biased *impact* – enables differentiated "importance" for multiple system output signals
  - Constant scaling of *impact* for single output systems
- ❖ Criticality of output signals assigned by system designer

Criticality  $C_{s,i}$  of signal  $S_s$  on output signal  $S_{o,i}$

$$0 \leq C_{s,i} = C_{o,i} \cdot (S_s \gg S_{o,i}) \leq 1$$

$C_{o,i}$  = Criticality of system output  $S_{o,i}$

Total Criticality  $C_s$  signal  $S_s$

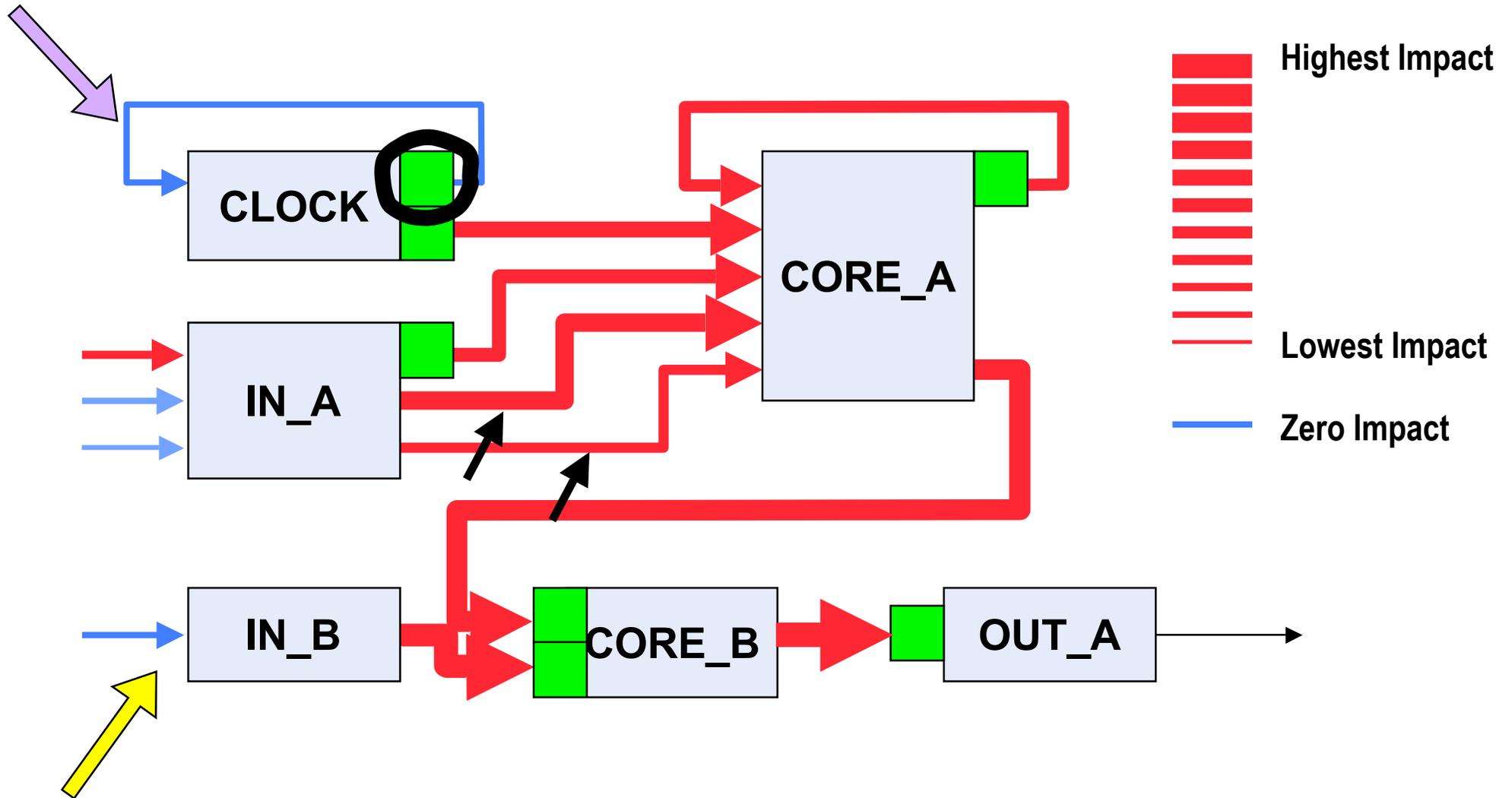
$$0 \leq C_s = 1 \prod_i (1 \leq C_{s,i})$$

## Guide to Interpreting Impact/Criticality

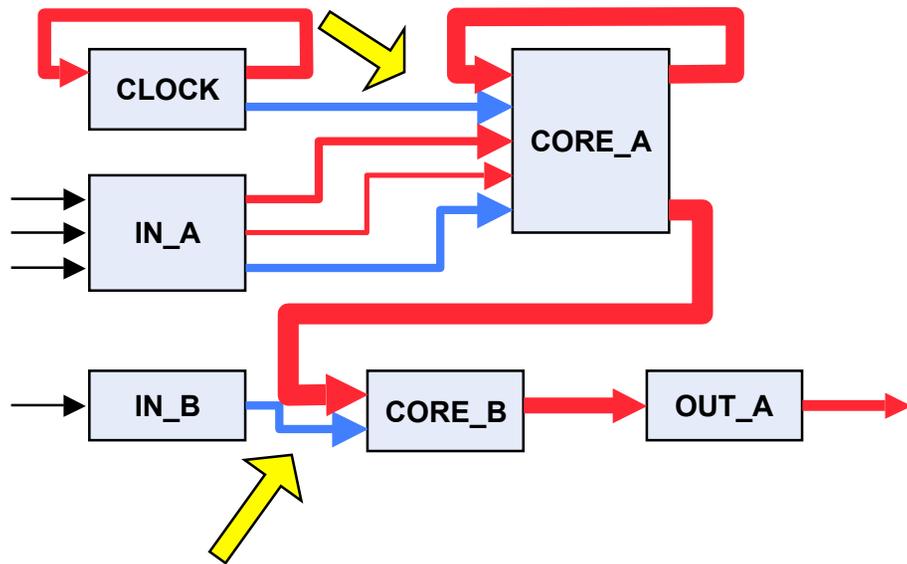
Higher *impact/criticality* implies higher probability of an error propagating beyond the system boundary and causing "expensive" damage: *error "cost" metric*

- Error containment may be increased even though the *exposure* is very low (or even zero)
- May increase system survivability to place **EDM's/ERM's** here

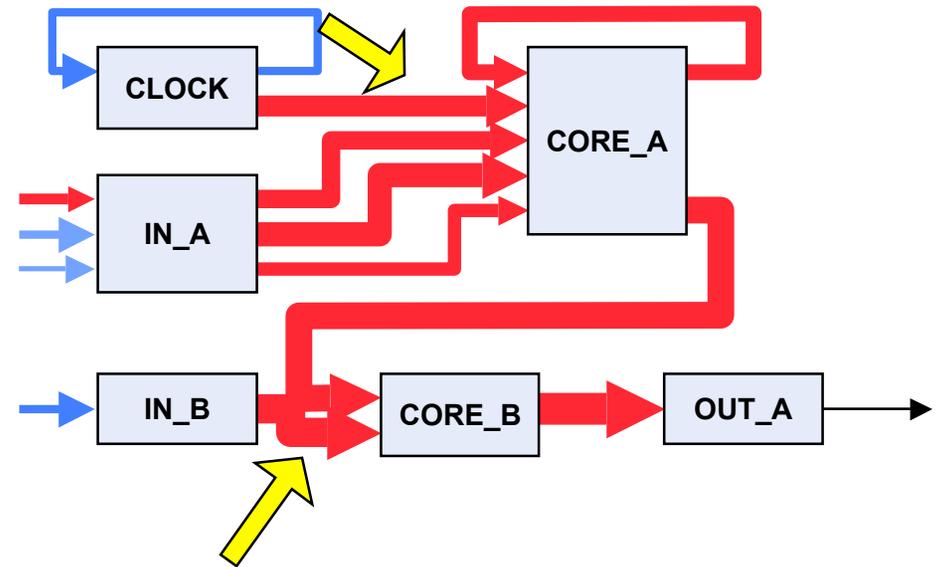
# Estimated Error Impact



# Exposure Vs. Impact: SW Profiling



**Exposure  
(Prop. Profile)**

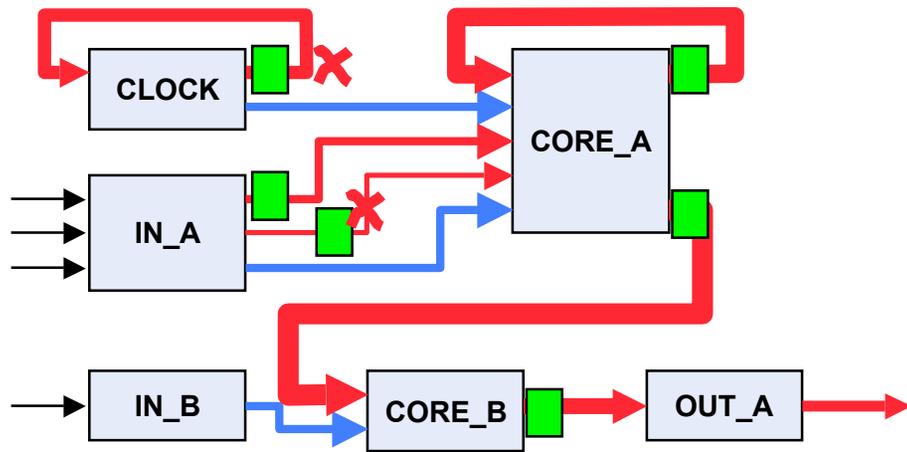


**Impact  
(Effect Profile)**

# EA Placement

**Assumptions and design information:**

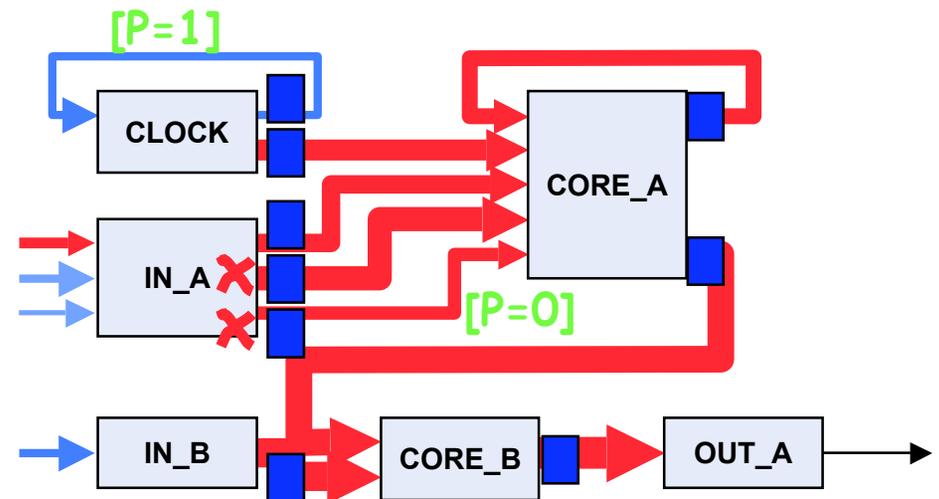
- 1. Trans. Errors are introduced via system inputs
- 2. Executable Assertions (not aimed at boolean values)
- 3. Black-box software



**Basic Prop./Exposure Profile**

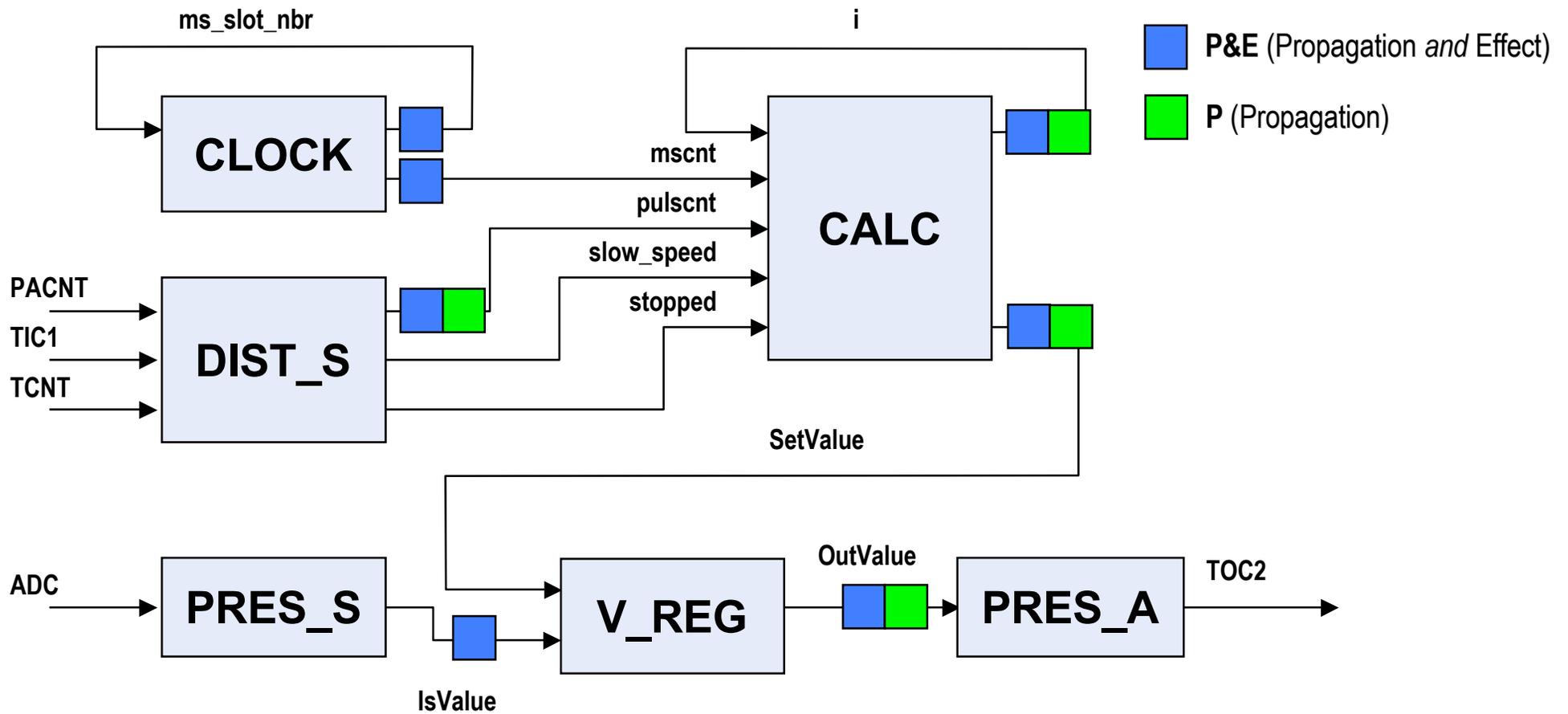
**Assumptions and design information:**

- 1. Trans. Errors are introduced anywhere in memory
- 2. Executable Assertions (not aimed at boolean values)
- 3. Black-box software

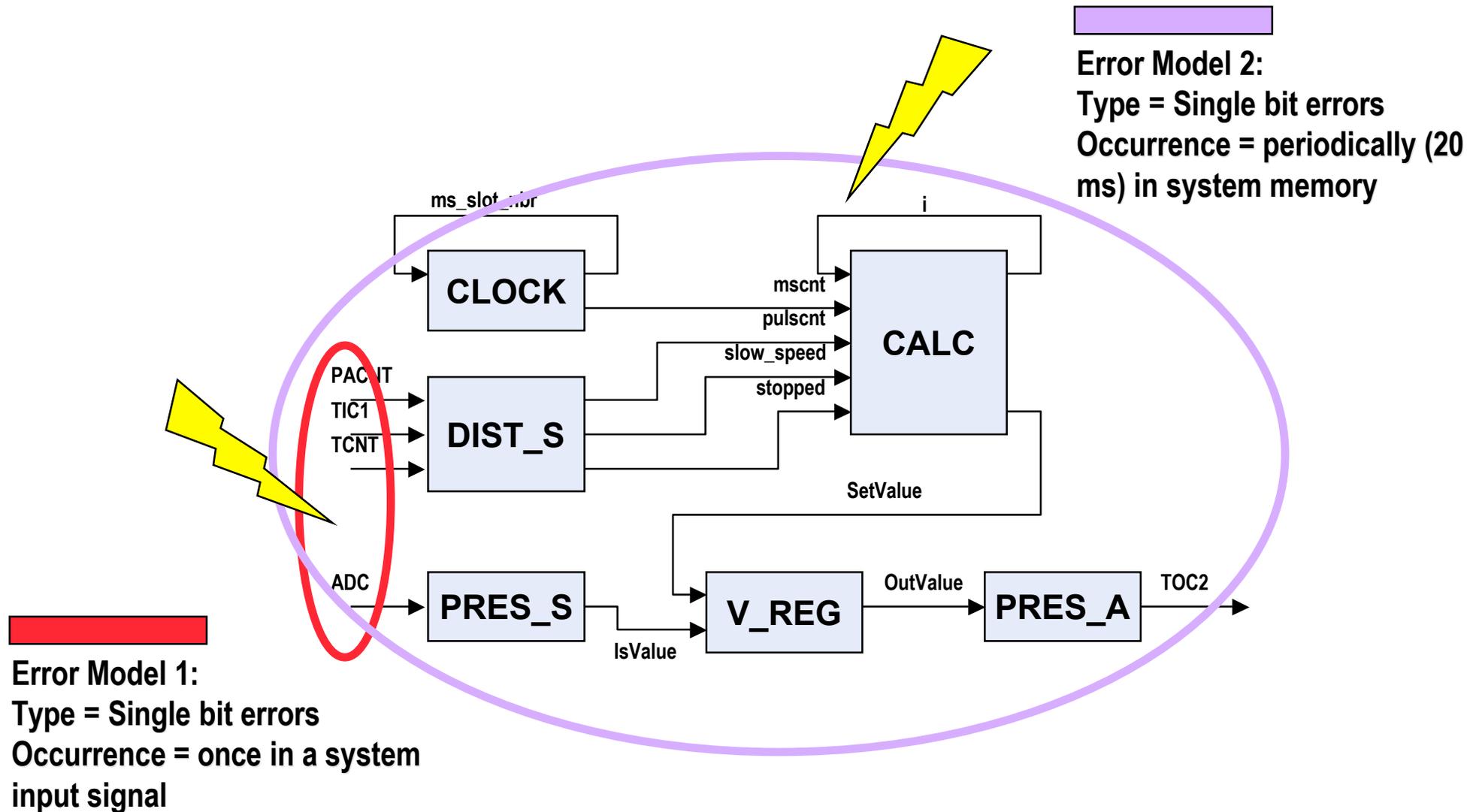


**Composite Prop. + Effect Profile**

# Two Sets of EA's



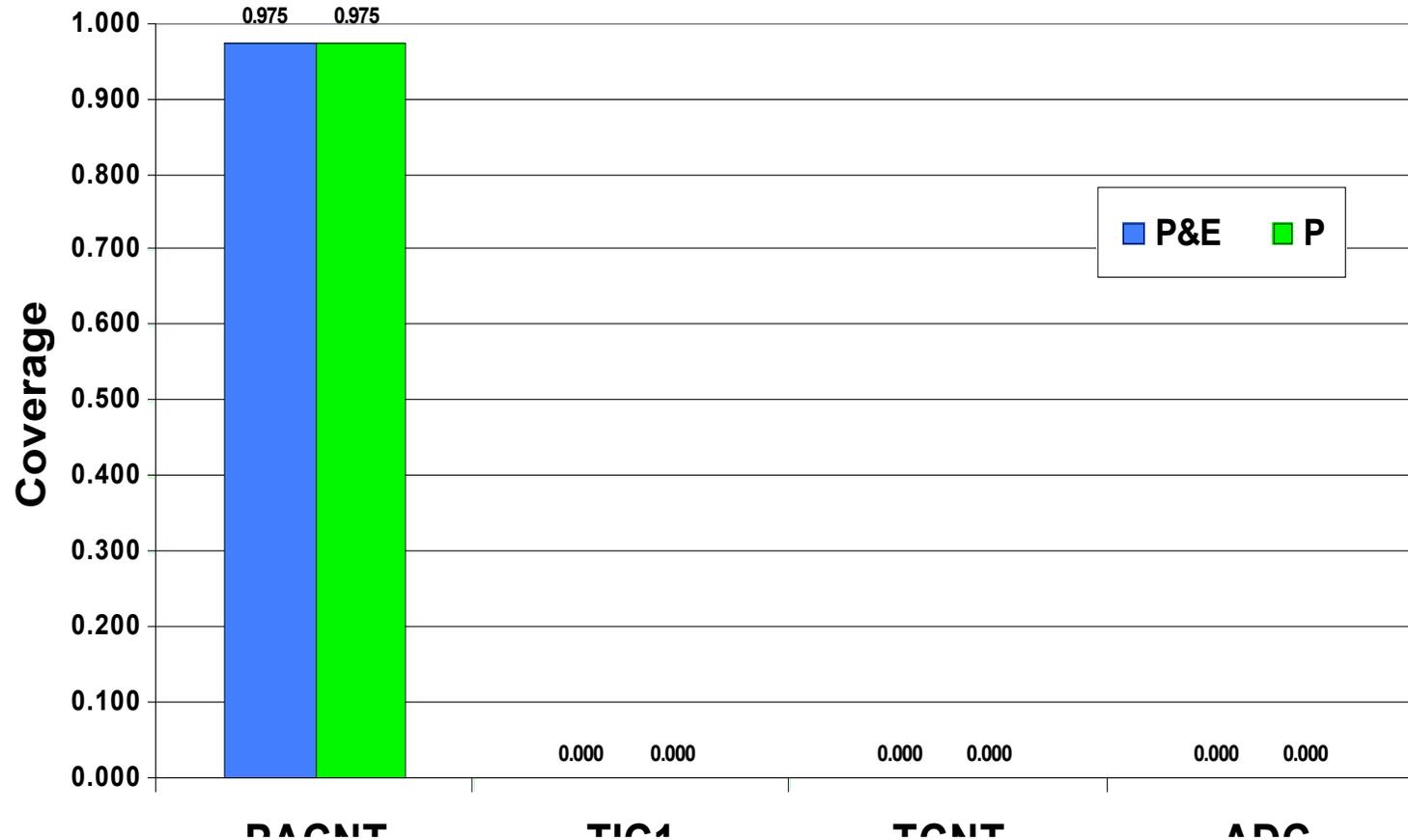
# Evaluation With Varied Error Models



# Detection Coverage - Error Model 1



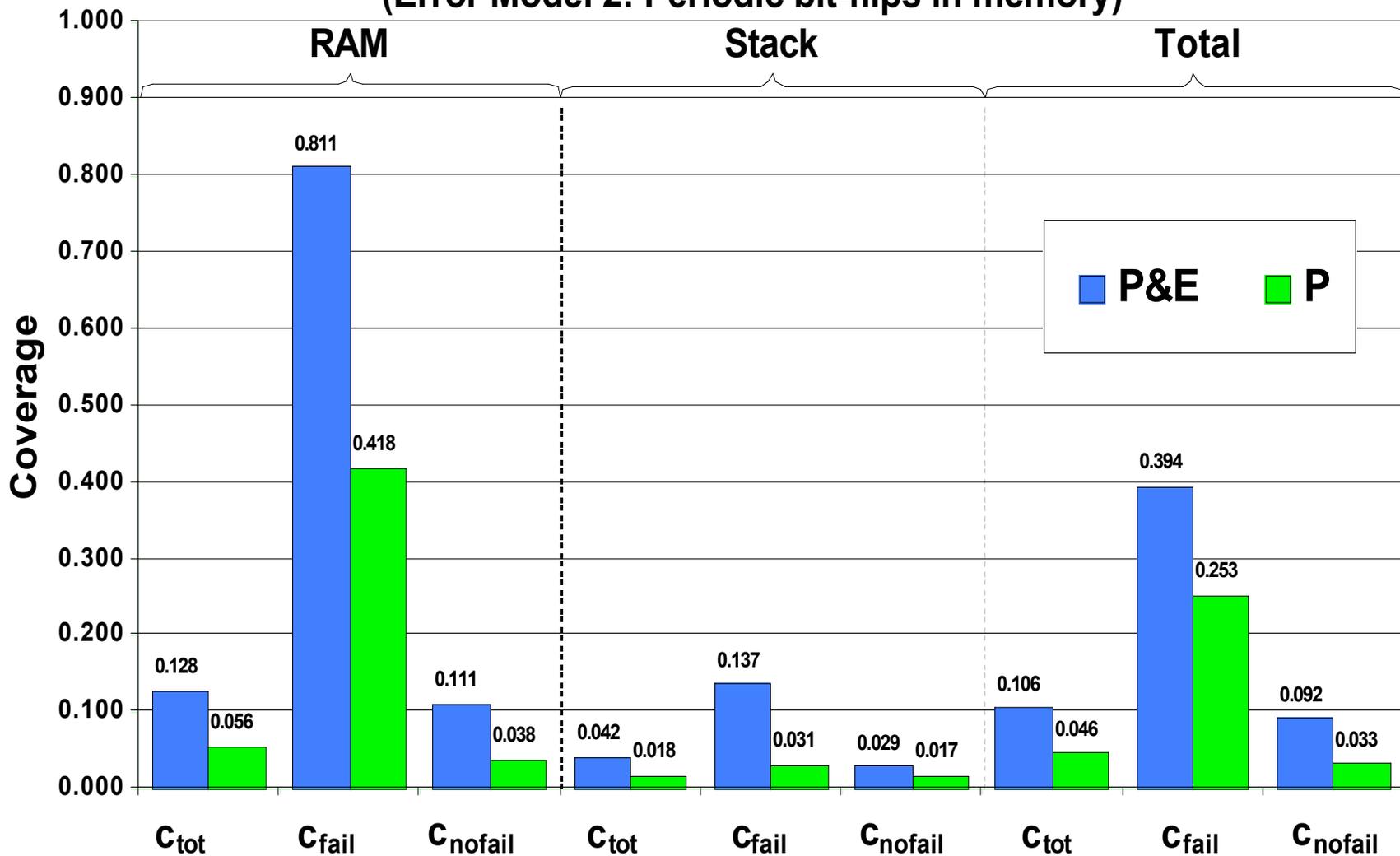
**Comparison of Error Detection Coverage  
(Error Model 1: Single bit-flips in system input signals)**



# Detection Coverage - Error Model 2



Comparison of Error Detection Coverage  
(Error Model 2: Periodic bit-flips in memory)



# Error Detection Probabilities

Signal	Measure	EA1	EA2	EA3	EA4	EA5	EA6	EA7
SetValue	P(d)	<b>55.5±4.1</b>	31.3±3.8	4.0±1.6				44.3±4.1
	P(d fail)	<b>92.6±3.7</b>	72.4±6.4	1.5±1.7				87.9±4.7
	P(d no fail)	<b>36.6±4.9</b>	10.5±3.1	5.3±2.3				22.8±4.2
IsValue	P(d)		<b>52.5±4.1</b>					47.0±4.1
	P(d fail)		<b>89.6±7.3</b>					93.3±6.2
	P(d no fail)		<b>47.4±4.4</b>					41.1±4.3
i	P(d)	26.8±3.6	29.8±3.8	<b>100.0</b>	1.5±1.0	1.0±0.8	0.5±0.6	47.8±4.1
	P(d fail)	33.7±7.8	55.4±8.2	<b>100.0</b>	2.0±2.3	2.3±2.1	1.1±1.8	78.0±6.8
	P(d no fail)	24.4±4.1	21.1±3.9	<b>100.0</b>	1.3±1.1	0.4±0.6	0.3±0.5	37.7±4.6
pulsent	P(d)	50.3±4.1	42.8±4.1	0.3±0.4	<b>12.8±2.7</b>			0.3±0.4
	P(d fail)	38.1±5.3	34.5±4.8	0.3±0.5	<b>0.0</b>			0.7±1.2
	P(d no fail)	66.9±6.0	58.3±6.9	0.0	<b>16.6±3.5</b>			0.0
ms_slot_nbr	P(d)		20.0±3.3			<b>100.0</b>		6.8±2.1
	P(d fail)		34.6±5.7			<b>100.0</b>		11.6±3.9
	P(d no fail)		7.1±2.9			<b>100.0</b>		2.7±1.8
mscnt	P(d)	8.3±2.3	12.3±2.7				<b>100.0</b>	17.5±3.1
	P(d fail)	20.0±13.4	18.2±13.8				<b>100.0</b>	13.0±11.8
	P(d no fail)	7.5±2.2	11.9±2.7				<b>100.0</b>	17.8±3.2
OutValue	P(d)		1.0±0.8					<b>11.3±2.6</b>
	P(d fail)		33.3±34.7					<b>85.7±23.5</b>
	P(d no fail)		0.5±0.6					<b>9.9±2.5</b>
<b>Total</b>	P(d)	20.1±1.2	27.1±1.4	14.9±1.1	2.0±0.4	14.4±1.1	14.4±1.1	25.0±1.3
	P(d fail)	35.0±2.9	47.0±3.0	12.2±1.9	0.3±0.4	21.7±2.3	3.2±1.0	42.7±3.3
	P(d no fail)	14.9±1.3	19.7±1.4	16.0±1.4	2.5±0.5	11.1±1.2	19.0±1.5	19.9±1.4

~ 100%

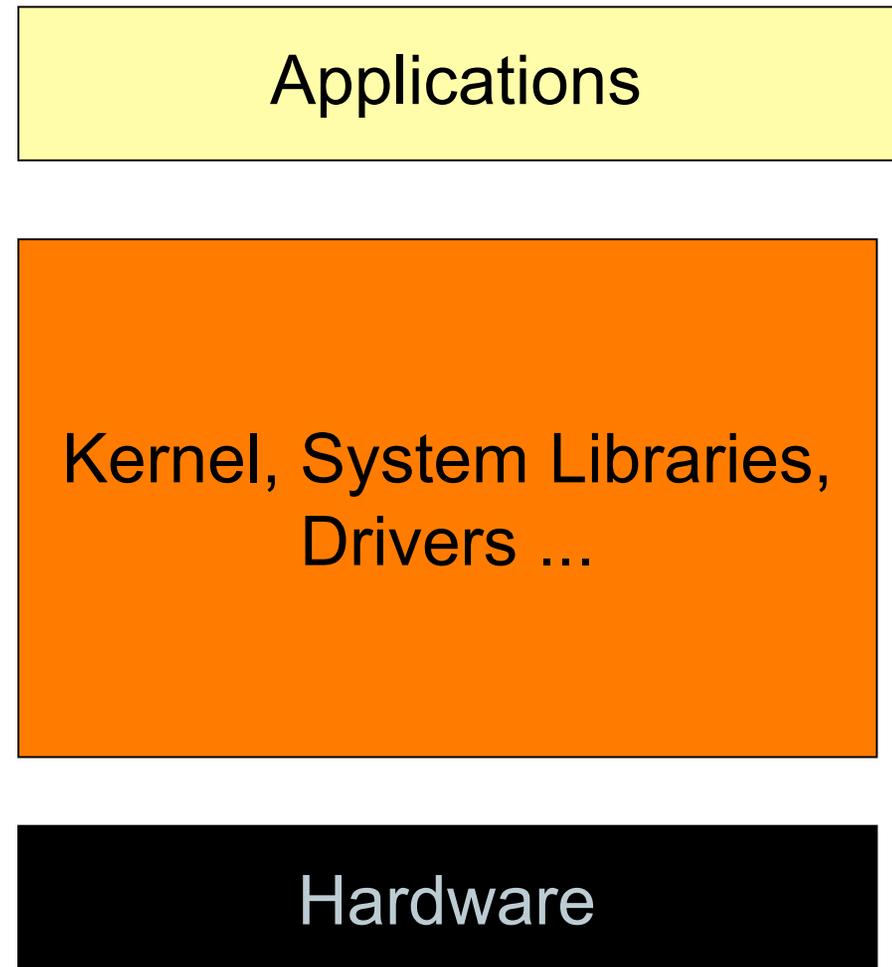
## So far...error propagation & effect profiling

- Method for software profiling
  - **Error propagation profile:** pinpoint vulnerable signals/modules and ascertain propagation paths (**Exposure, Permeability**)
  - **Error effect profile:** pinpoint signals/modules that endanger the system when erroneous (**Impact, Criticality**)
- Error model and detection coverage
  - Different SW profiles for different error models
  - ▷ **Design aid to conduct cost-benefit analysis for selective placement**

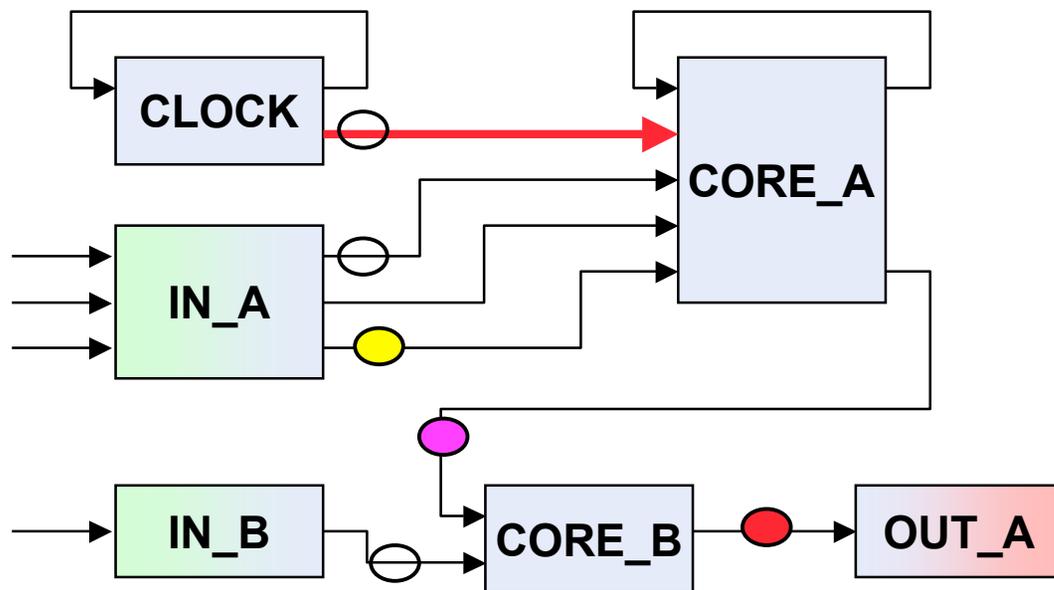
## Going onwards I: Static vs. Dynamic Profiling

1. for the OS designer – to locate potential hot spots related with errors
2. for the applications designer – to evaluate how the application will be affected by an error at OS level
3. run-time wrapper formulation and placement!

OS



## Going onwards II ... localized vs. distributed EA's

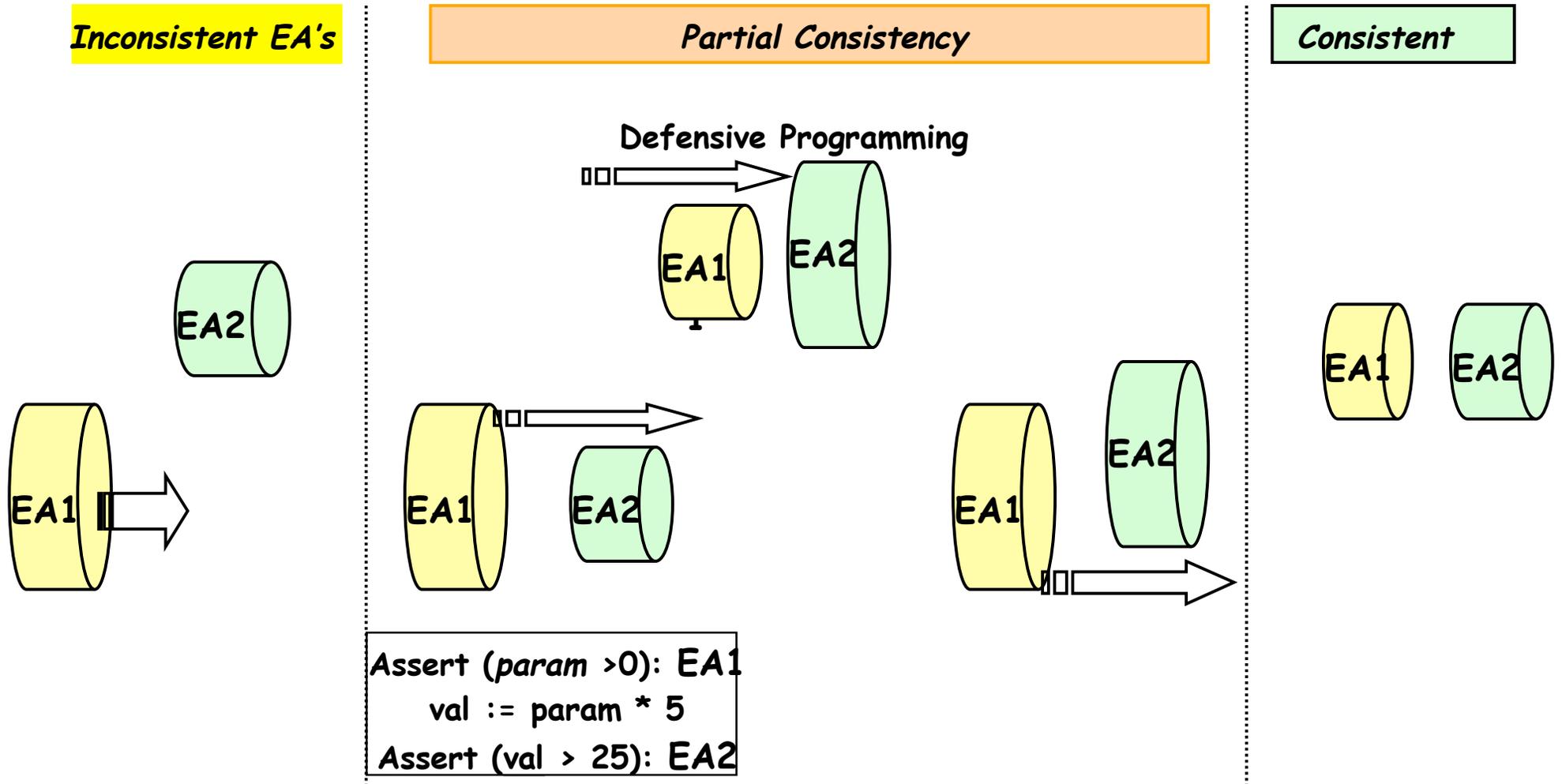


... so far the "effectiveness" of EA placement has been a localized process

❖ Do localized EA's add up to implement global EA's? What complementary aspects across EA's can be useful?

❖ Can composite/global EA designs be combined with EA placement aspects?

# Conformal (Consistency) aspects across EA's (Filters)?

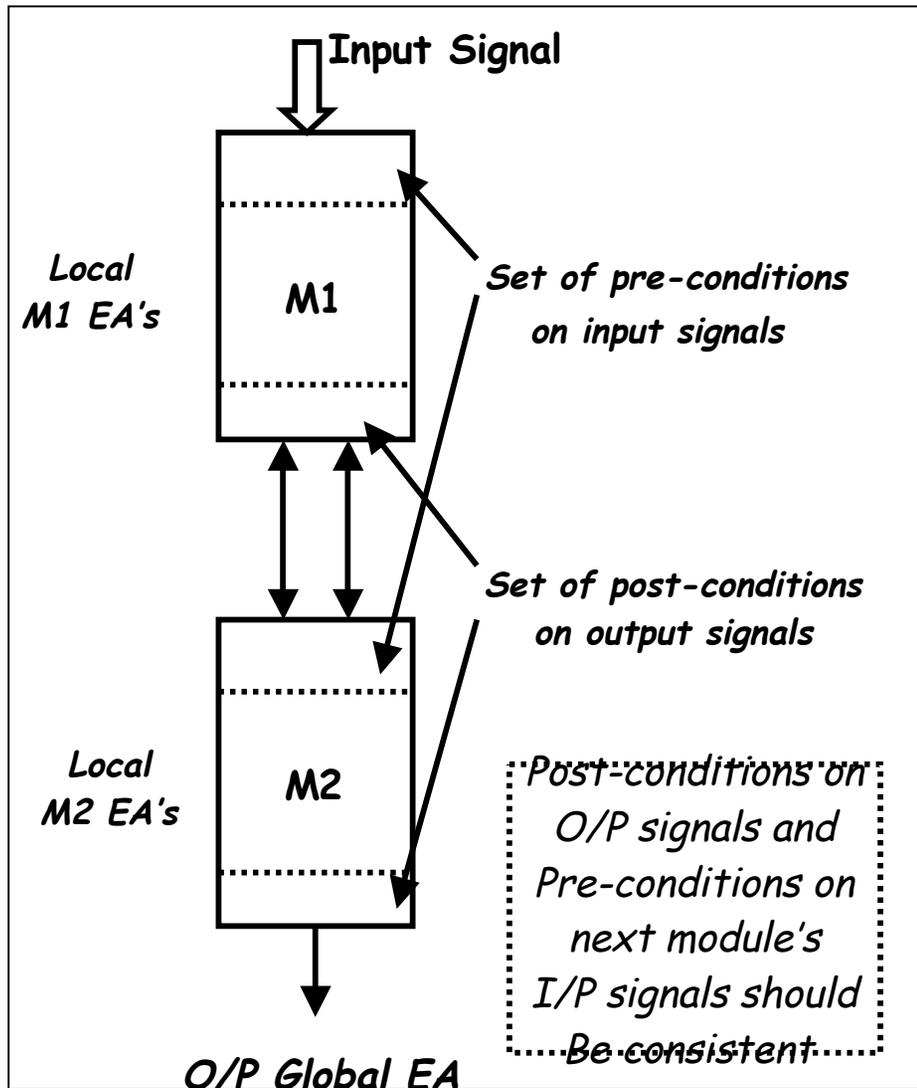


- Semantics based framework for specification & verification of EA's for consistency (in data values)
- Run time compile support for EA refinement for inter-EA consistency

**Outlook:** inter-EA consistency or inconsistency can highlight conformal vulnerabilities and limits for EA coverage expectations

optimality for EA constitution/placement in some sense?

# Inter-EA Consistency Check + Provisioning



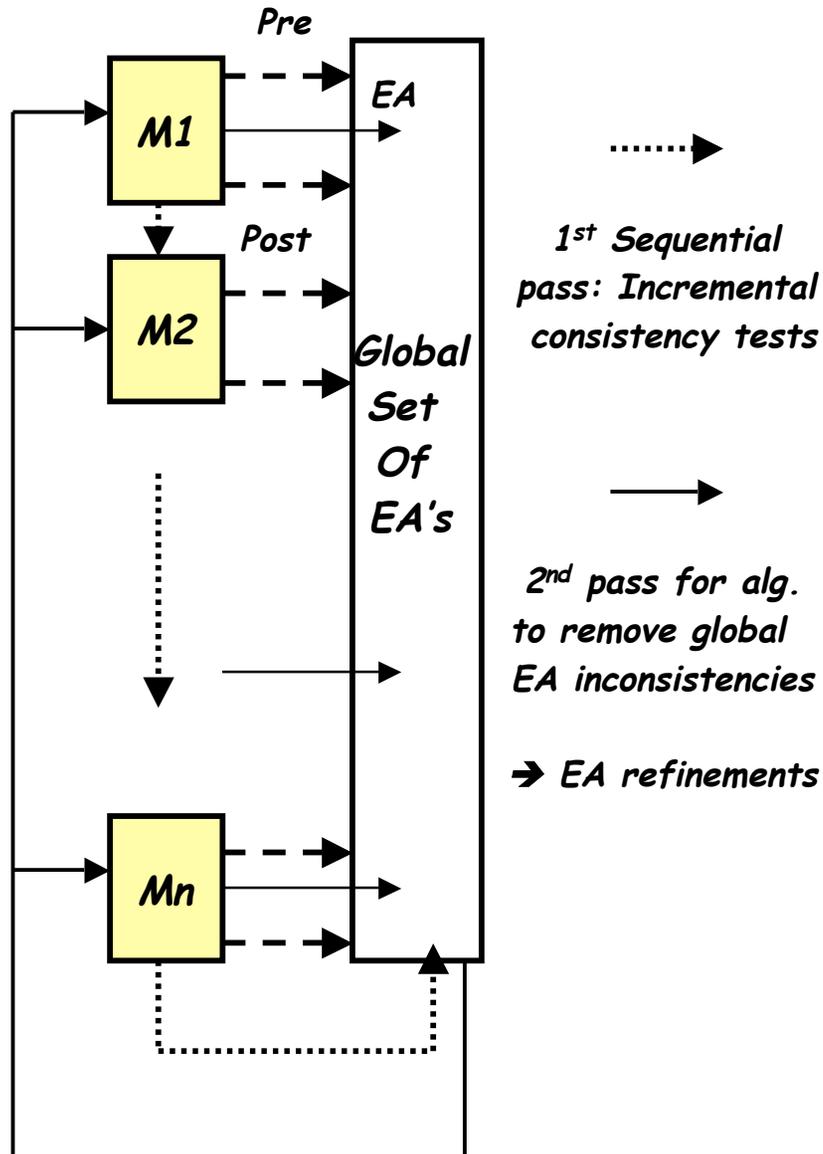
Formalized EA specs:

Assert, annotations (APP)

Instead of setting up and proving verification conditions, code is generated to check whether pre and post-conditions are satisfied at run-time

(avoids some of the aspects of tractability for loops and unwieldy assertions that hits verification)

# Inter-EA Consistency



Static Analysis: Compiler level checks + verification using annotations etc

Backtrack analysis + refinement  
Gcc level support

## Synopsis

- Systematizing SW error propagation profiling w.r.t. permeability, exposure, impact/cost ... error containment + aid for designer level tradeoffs
- Inter-EA consistency: vulnerability analysis, ...
- ❖ Optimality of EA placement (exposure, impact, criticality) + inter-EA conformity (value & time!)??
- ❖ Embedded OS Robustness wrapping

## Some Future Directions

- Alternate means of estimating error permeability (dynamically or statically)
- Sensitivity of estimated measures to error model
- Handling looping structures amongst modules
  
- Dynamic effect propagation and profiling!
- On-the-fly wrapper composition for changing dependability requirements

[www.deeds.informatik.tu-darmstadt.de](http://www.deeds.informatik.tu-darmstadt.de)